INTRODUCTION:

An operating system is a program that manages a computer's hardware. It also provides a basis for application programs and acts as an intermediary between the computer user and the computer hardware. An amazing aspect of operating systems is how they vary in accomplishing these tasks. Mainframe operating systems are designed primarily to optimize utilization of hardware. Personal computer (PC) operating systems support complex games, business applications, and everything in between. Operating systems for mobile computers provide an environment in which a user can easily interface with the computer to execute programs. Thus, some operating systems are designed to be convenient, others to be efficient, and others to be some combination of the two. Before we can explore the details of computer system operation, we need to know something about system structure. We thus discuss the basic functions of system startup, I/O, and storage early in this chapter. We also describe the basic computer architecture that makes it possible to write a functional operating system. Because an operating system is large and complex, it must be created piece by piece. Each of these pieces should be a well-delineated portion of the system, with carefully defined inputs, outputs, and functions. In this chapter, we provide a general overview of the major components of a contemporary computer system as well as the functions provided by the operating system. Additionally, we cover several other topics to help set the stage for the remainder of this text: data structures used in operating systems, computing environments, and open-source operating systems

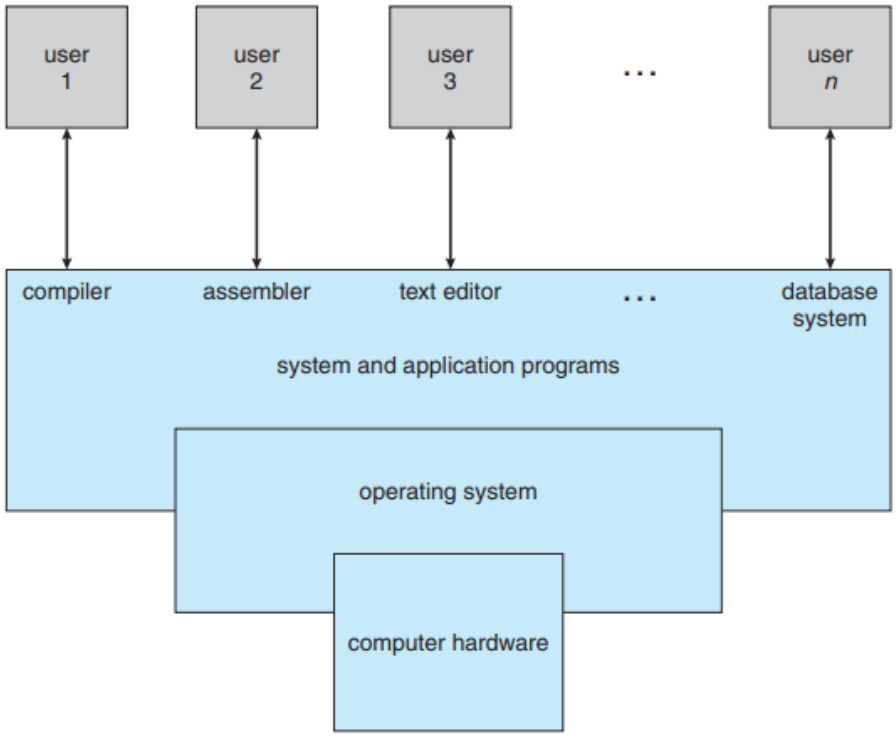

Figure 1.1 Abstract view of the components of a computer system.

1.1.3 Defining Operating Systems

An OS encompasses many roles due to the diverse designs and uses of computers, which range from toasters to mainframes. Computers have evolved from experimental tools to general-purpose systems, leading to the development of operating systems to manage hardware and application programs efficiently.

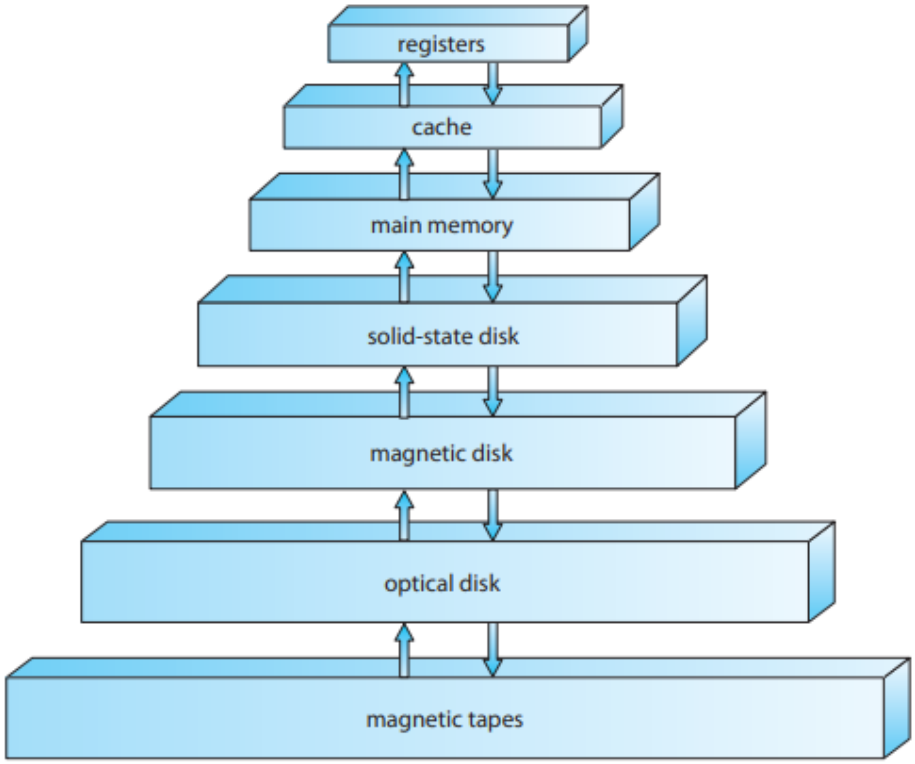

Figure 1.4 Storage-device hierarchy.

1.2.2 Storage Structure

The CPU loads instructions from memory, so programs must be stored in main memory (RAM), which is typically implemented as dynamic random-access memory (DRAM). Other types of memory include read-only memory (ROM) and electrically erasable programmable read-only memory (EEPROM). ROM stores unchangeable static programs, while EEPROM, although modifiable, is mainly used for static programs due to its infrequent change capability.

Memory is organized as an array of bytes, each with a unique address, accessed through load or store instructions. The CPU fetches instructions from memory, decodes them, and executes them, potentially storing results back in memory.

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Figure 1.11 Performance of various levels of storage.

Process Control Block:

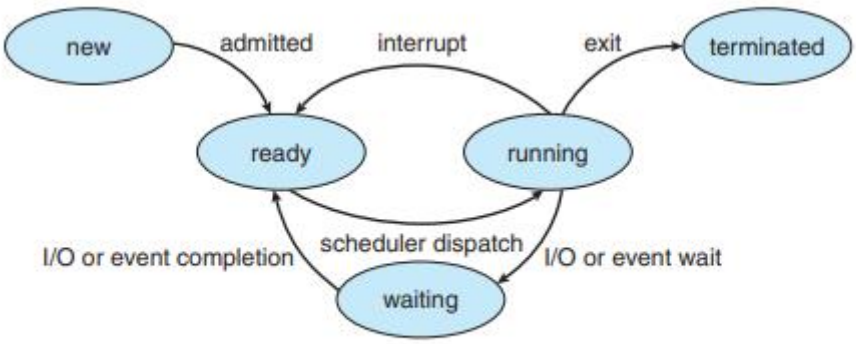

Figure 3.2 Diagram of process state.

Each process is represented in the operating system by a process control block (PCB)—also called a task control block. A PCB is shown in Figure

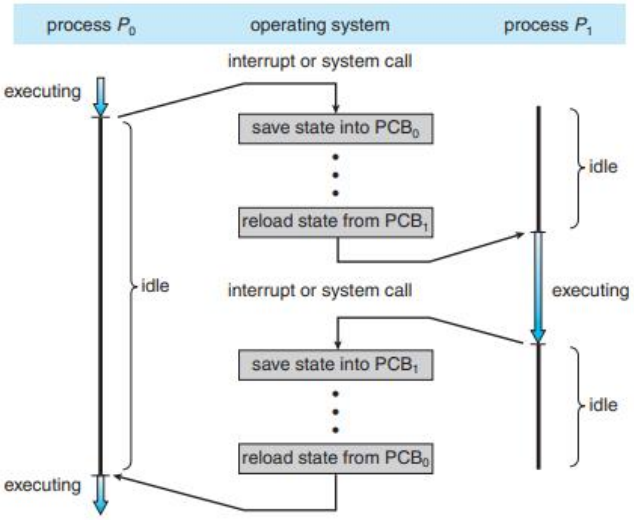

Figure 3.4 Diagram showing CPU switch from process to process.

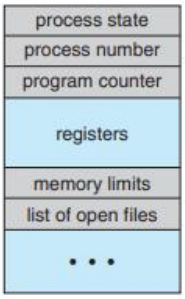

Figure 3.3 Process control block (PCB).